IN THE CLAIMS

1. (Currently Amended) A method comprising:

identifying scratch values generated during speculative execution of a processor; ~~and~~

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value; and

bypassing execution of instructions having at least one operand with an associated tag that indicates that the operand is a scratch value.

2. (Original) The method of claim 1 wherein setting comprises setting the tag of a register when an instruction having the register as a destination results in a cache miss.

3. (Previously Presented) The method of claim 1 further comprising:

propagating the tag to destination registers of the instructions to indicate that an operand within the destination registers is a scratch value.

4. (Previously Presented) The method of claim 1 further comprising:

bypassing execution of an arithmetic instruction having at least one register as an operand with an associated tag indicating that the register contains data that is a scratch value; and

bypassing execution of a store instruction involving a value derived from a register having an associated tag indicating that the register contains data that is a scratch value.

5. (Original) The method of claim 1 further comprising:

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

6. (Original) The method of claim 1 further comprising:

marking each instruction in a pipeline with a tag to indicate if the instruction involves a scratch value.

7. (Original) The method of claim 1 further comprising:

propagating the tag through a store buffer if an address generation register does not indicate that the address generation register holds a scratch value.

8. (Previously Presented) A processor comprising:

a plurality of registers having a corresponding plurality of register tags to indicate whether the data stored in the register holds a scratch value;

a plurality of flags having a corresponding plurality of flag tags to indicate whether the data reflected by the flag is based on a scratch value;

a plurality of predicates having a corresponding plurality of predicate tags to indicate whether the data reflected by the predicate is based on a scratch value; and

an execution engine to bypass execution of instructions having at least one operand with an associated tag that indicates that the operand is a scratch value.

9. (Original) The processor of claim 8 having an instruction set including a plurality of instructions, each instruction augmented by an instruction tag to indicate whether the instruction involves a scratch value.

10. (Original) The processor of claim 9 wherein the register tags, flag tags, predicate tags, and instruction tags have a size of one bit.

11. (Original) The processor of claim 9 wherein the register tags, flag tags, predicate tags and instruction tags have a size of at least two bits.

12. (Previously Presented) A processor comprising:

at least two arithmetic units;

a translation look aside buffer;

a branch prediction unit; and

an execution engine having a plurality of instructions which when executed cause the processor to perform actions including:

identifying scratch values generated during speculative execution of a processor,

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value; and

bypassing execution of instructions having at least one operand with an associated tag that indicates that the operand is a scratch value.

13. (Original) The processor of claim 12 wherein setting comprises setting the tag of a register when an instruction having the register as a destination results in a cache miss.

14. (Previously Presented) The processor of claim 12 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

propagating the tag to destination registers of the instructions to indicate that the operand within the destination registers is a scratch value.

15. (Previously Presented) The processor of claim 12 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

bypassing execution of an arithmetic instruction having at least one register as an operand with an associated tag indicating that the register contains data that is a scratch value; and

bypassing execution of a store instruction involving a value derived from the register having an associated tag indicating that the register contains data that is a scratch value.

16. (Original) The processor of claim 12 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

17. (Original) The processor of claim 12 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

marking each instruction in a pipeline with a tag to indicate if the instruction involves a scratch value.

18. (Original) The processor of claim 12 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

propagating the tag through a store buffer if an address generation register does not indicate that the address generation register holds a scratch value.

19. (Previously Presented) A system comprising:

a memory, a storage device, and a processor each coupled to a bus;

the processor including an execution engine having instructions which when executed by the processor cause the processor to perform actions including:

identifying scratch values generated during speculative execution of a processor;

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value; and

bypassing execution of instructions having at least one operand with an associated tag that indicates that the operand is a scratch value.

20. (Original) The system of claim 19 wherein setting comprises setting the tag of a register when an instruction having the register as a destination results in a cache miss.

21. (Previously Presented) The system of claim 19 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

propagating the tag to destination registers of the instructions indicate that an operand within the destination registers is a scratch value.

22. (Previously Presented) The system of claim 19 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

bypassing execution of an arithmetic instruction having at least one register as an operand with an associated tag indicating that the register contains data that is a scratch value; and

bypassing execution of a store instruction involving a value derived from the register having an associated tag indicating that the register contains data that is a scratch value.

23. (Original) The system of claim 19 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

24. (Original) The system of claim 19 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

marking each instruction in a pipeline with a tag to indicate if the instruction involves a scratch value.

25. (Original) The system of claim 19 wherein the execution engine has further instructions which when executed cause the processor to perform further actions comprising:

propagating the tag through a store buffer if an address generation register does not indicate that the address generation register holds a scratch value.

26. (Previously Presented) A machine readable medium having instructions stored thereon which when executed by a processor cause the processor to perform actions including:

identifying scratch values generated during speculative execution of a processor;

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value; and

bypassing execution of instructions having at least one operand with an associated tag that indicates that the operand is a scratch value.

27. (Original) The machine readable medium of claim 26 wherein setting comprises setting the tag of a register when an instruction having the register as a destination results in a cache miss.

28. (Previously Presented) The machine readable medium of claim 26 having further instructions which when executed cause the processor to perform further actions comprising:

propagating the tag to destination registers of the instructions to indicate that an operand within the destination registers is a scratch value.

29. (Previously Presented) The machine readable medium of claim 26 having further instructions which when executed cause the processor to perform further actions comprising:

bypassing execution of an arithmetic instruction having at least one register as an operand with an associated tag indicating that the register contains data that is a scratch value; and

bypassing execution of a store instruction involving a value derived from the register having an associated tag indicating that the register contains data that is a scratch value.

30. (Original) The machine readable medium of claim 26 having further instructions which when executed cause the processor to perform further actions comprising:

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

31. (Currently Amended) A method comprising:

identifying scratch values generated during speculative execution of a processor;~~ and~~

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value; and

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

32. (Previously Presented) A processor comprising:

at least two arithmetic units;

a translation look aside buffer;

a branch prediction unit; and

an execution engine having a plurality of instructions which when executed cause the processor to perform actions including:

identifying scratch values generated during speculative execution of a processor,

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value, and

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

33. (Previously Presented) A system comprising:

a memory, a storage device, and a processor each coupled to a bus;

the processor including an execution engine having instructions which when executed by the processor cause the processor to perform actions including:

identifying scratch values generated during speculative execution of a processor,

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value, and

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

34. (Previously Presented) A machine readable medium having instructions stored thereon which when executed by a processor cause the processor to perform actions including:

identifying scratch values generated during speculative execution of a processor;

setting at least one tag associated with at least one data area of the processor to indicate that the data area holds a scratch value; and

utilizing a branch predictor to override computed branch results produced by branch instructions based on data having a tag that indicates the data is a scratch value.

35. (Previously Presented) A method comprising:

setting a tag of a register when an instruction having the register as a destination results in a cache miss, to identify the register as containing a scratch value;

bypassing execution of instructions having at least one operand with an associated tag that indicates that the operand is a scratch value until at least one instruction is detected that results in a cache miss; and

re-executing bypassed instructions once a cache is loaded with cache miss data.

36. (Previously Presented) The method of claim 35, further comprising:

executing store instructions having a tag to indicate that the instruction involves a scratch value if tag values associated with operands of the store instruction indicate non-scratch values.

37. (Previously Presented) The method of claim 35, further comprising:

propagating the tag to destination registers of the instructions to indicate that an operand within the destination registers is a scratch value.

38. (Previously Presented) The method of claim 35, further comprising:

servicing detected cache miss instructions in parallel prior to re-executing the bypassed instructions.